

IN THE UNITED STATES OF AMERICA PATENT AND TRADEMARK OFFICE

UNITED STATES OF AMERICA PATENT APPLICATION

FOR:

APPARATUS, METHOD AND SYSTEM FOR A TEMPORAL INTERFACE,
INTERPRETIVE HELP, DIRECTED SEARCHES, AND DYNAMIC ASSOCIATION
MAPPING

INVENTOR:

KONATA STINSON

Morgan & Finnegan, L.L.P.

345 Park Avenue
New York, New York 10154-0053
United States of America
(212) 758-4800

Attorneys for Applicant

596560 v.1

EL 762530722US

**APPARATUS, METHOD AND SYSTEM FOR A TEMPORAL INTERFACE,
INTERPRETIVE HELP, DIRECTED SEARCHES, AND DYNAMIC ASSOCIATION
MAPPING**

This application claims priority under 35 USC §
5 119(e) to provisional application serial no. 60/176,470, filed
January 17, 2000 and provisional application serial no.
60/176,614, filed January 18, 2000, each of which are
incorporated herein by reference.

FIELD

10 The present invention generally relates to computer
software, devices and methods, and more particularly to
computer interfaces, search, help, and relational facilities.

BACKGROUND

15 A wide variety of machinery, systems, and methods
exist, which allow for the interaction, viewing, mapping, and
searching of information systems. The proliferation and
expansion of computer systems and the Internet, and
particularly the World Wide Web (WWW), has resulted in a vast
and diverse collection of information organized as hypertext.
20 Various interaction interfaces (user interfaces) exist that

facilitate the interaction of people with computer systems.

Graphical user interfaces provided by likes of the Apple Macintosh Operating System, or Microsoft's Windows 98, provide a baseline and means of accessing and displaying information.

5 Also, the advance of the WWW has provided information navigation interfaces, also known as web browsers, such as Microsoft's Internet Explorer and Netscape Navigator, as well as navigation interface devices such as WebTV. Many of these user interfaces also have context sensitive help system
10 facilities customarily, which are accessed through menu selections, keyboard entries, and allow for searches. Examples of these interfaces include the Microsoft Windows Help System, Apple's Macintosh Operating System balloon help, and popup help systems.

15 Information on the WWW is encoded in HyperText Markup Language (HTML); HTML documents are also commonly referred to as web pages. HTML documents may contain links to other HTML documents that can be traversed by users of graphical user interfaces and web browsers by selecting the
20 links, which are commonly highlighted by color and underlining. Web browsers, originally developed on NeXT

Computer Inc.'s operating system NeXTSTEP and now widely

available on almost every operating system platform, allow
users to access uniquely identified documents in the WWW by
entering a navigation location in a Universal Resource Locator
5 (URL) facility. The URL is the address or navigation location
for a resource accessible on the Internet. The basic paradigm
presents users with a scrolling page full of text, pictures,
and various other forms of information media such as movies,
and links to other documents. The vastness of the WWW has
10 resulted in the proliferation of web search engines such as
Yahoo, AltaVista, and Google to facilitate the finding of
relevant information. These web search engines take search
requests entered into a web page, or user interface
intermediary like Apple's Sherlock, and typically return links
15 (i.e., navigation locations) that the user may traverse.

Various attempts have been made to improve search
engine strategies. One such strategy is known as the fish
search which is discussed in greater detail in: dr. P.M.E. De
Bra & drs. R.D.J. Post, Searching for Arbitrary Information in
20 the WWW: the Fish-Search for Mosaic (visited January 13, 2000)
www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Searching/debra/articl

09761617 011601

e.html. Another similar search strategy is known as the shark search wish is discussed in greater detail in: Michael Hersovici, et al., The shark-search algorithm—An application: tailored Web site mapping (visited January 13, 2000)

5 www7.scu.edu.au/programme/fullpapers/1849/com1849.htm.

Data mapping interfaces have been improved providing interfaces to better examine and maneuver through the WWW such as: IBM's Mappaccino, Michael Hersovici, et al., The shark-search algorithm—An application: tailored Web site mapping
10 (visited January 13, 2000)

www7.scu.edu.au/programme/fullpapers/1849/com1849.htm; Nestor, Romain Zeiliger, Claire Belisle, et al., Implementing a Constructivist Approach to Web Navigation support (visited January 13, 2000) www.irpeacs.fr/~zeiliger/artem99.htm; and
15 Natto, The Natto View (visited January 13, 2000) www.myo.inst.keio.ac.jp/NattoView/.

SUMMARY

As set forth below, a need exists for an improved apparatus, method, and system for allowing users to access and
20 traverse wide and diverse collections of information. The user interfaces that allow for the traversing the WWW have

left users confused and overwhelmed with the resulting

displays, formation, and provisions for traversing.

The proliferation of information has stressed the functionality of current user interfaces. The sheer magnitude

5 of interrelated information, offered on the WWW alone, is staggering. Old user interface paradigms such as the desktop metaphor provided by Microsoft Windows were sufficient to

handle the access and navigation of locally stored

information. However, with the advent and proliferation of

10 network communications, such desktop paradigms have been

stressed. The easy location, navigation and access of a few

documents maps well to a desktop metaphor. However, even with

relatively modest sized Local Area Networks (LANs), the

ability to easily find, access, and navigate through data has

15 been hampered to the point where new paradigms of information

access were developed. Web browsers presented a user

interface that allowed users to access information with less

regard to physical location. However, the sheer vastness of

information on the Internet and its distributed and

20 decentralized nature makes it exceedingly difficult to find

relevant and particular documents, save and manage navigation

locations of interest, or navigate about related, relevant

information of interest to the user. For example, in Netscape

Navigator, the user is not provided with a visual

representation of their location in context with other related

5 locations and materials. If a user performs a search in a web

search engine such as AltaVista, many outdated and irrelevant

documents are returned. Along with the sheer amount of info

available, its dynamic nature (i.e., Web sites and pages

constantly being created, moved or removed) hampers

10 traditional search engines. When the user does find several

relevant documents, there is no easy way to save or view the

group of documents (or the subweb); to accomplish this, the

user must create a category in their bookmarks facility and

access and save each document of interest, thus, burdening

15 them with this menial organizational task.

Projects such as IBM's Mappaccino, Natto, and others

of the like have sought to improve mapping, searching, access,

and navigation of information with limited success. The

interfaces provided by the aforementioned have been confusing,

20 cumbersome, and unintuitive. They produce very crowded

directed graphs that provide few cues as to what information

may be found or useful at the represented navigation location,

typically in incredibly cluttered views. Even going into a third dimension with projects such as Natto have not eased matters. There has been no tight, visceral, and dynamic

5 integration and representation of information, mapping, or navigation presented to the user for such data access until the present invention.

The present invention provides a radical new paradigm in allowing the user to interact and access this enormous base of information. The fundamental premise is to provide a temporal interface to facilitate the navigation of information. The view presented to the user is that of a (liquid) data pool either atop their desktop interface, in a separate window, or in a subview of a window or the like.

10
15 Into this pool (in one embodiment, represented as water) information rains. The raindrops disturb the water similarly to how such natural phenomena occurs causing ripples. The raindrops in proximity to one another represent related information to the user. If a user does not interact with a
20 raindrop, the information fades with time and the water calms as the raindrops dissolve until new raindrops appear.

This pool allows the easy representation of vast, fluid, and complex mappings such as WWW and its subwebs. This user interface paradigm in conjunction with more intelligent dynamic analysis and mapping techniques gives a user a facility to easily and more readily access, find, and navigate data. Dynamic analysis tools such as the shark search have been greatly stunted by limited analysis techniques and a fundamental barrier in the assimilation of information to be presented in meaningful manners.

Finally, the present invention further facilitates user interaction with data by advancing the art of help systems. The current state of the art help is problematic to many users in that it requires the user to know a priori that which she needs help with. For example, when searching the help system in Microsoft Windows, one must guess the proper term or recognize the term in an index. This facility has been somewhat extended to context sensitive help. The context sensitive help allows the user to engage a modifier key, commonly the F1 key, and select a user interface element, which will bring up a related help file. This is problematic when the user is trying to figure out what she is doing wrong

because it does not interpret the user's actions. Rather, the user must fumble about the screen making selections hoping to guess the right element to select for help. The present invention analyzes and interprets the user's actions to guide them to a proper source for help and allows them to undo any mistakes they may have made.

The above advantages and features are of representative embodiments only, and are not exhaustive and/or exclusive. They are presented only to assist in understanding the invention. It should be understood that they are not representative of all the inventions defined by the claims, to be considered limitations on the invention as defined by the claims, or limitations on equivalents to the claims. For instance, some of these advantages may be mutually contradictory, in that they cannot be simultaneously present in a single embodiment. Similarly, some advantages are applicable to one aspect of the invention, and inapplicable to others. Furthermore, certain aspects of the claimed invention have not been discussed herein. However, no inference should be drawn regarding those discussed herein relative to those not discussed herein other than for purposes of space and

reducing repetition. Thus, this summary of features and advantages should not be considered dispositive in determining equivalence. Additional features and advantages of the invention will become apparent in the following description, from the drawings, and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings illustrate certain embodiments of the invention.

Fig. 1 illustrates a centralized controller according to one embodiment of the present invention;

Fig. 2 illustrates another embodiment of the present invention in the form of a distributed system interacting through a communications network;

Fig. 3 illustrates another embodiment of the system and various interactions;

Fig. 4 illustrates web pages, hypertext, reference and proximal links. 2;

Fig. 5 is a flowchart illustrating dynamic analysis mapping;

Fig. 6 is a flowchart illustrating a directed search;

Fig. 7 is a flowchart illustrating a dynamic directed search;

Fig. 8 is a flowchart illustrating an interpretive help facility;

5 Fig. 9 is an illustration of a temporal user interface;

Fig. 10 is a close up illustration of a temporal user interface;

10 Fig. 11 is an illustration of a temporal user interface with a selection being made;

Fig. 12 is an illustration of a temporal user interface focusing on information;

Fig. 13 is an illustration of a temporal user interface focusing on information;

15 Fig. 14 is an illustration of a temporal user interface windowfying on information;

Fig. 15 is an illustration of a temporal user interface deleting raindrops;

20 Fig. 16 is an illustration of a temporal user interface navigating through data by time;

Fig. 17 is an illustration of a temporal user

interface with a skimming pebble;

Fig. 18 is a flowchart illustrating the display flow
of a temporal user interface; and

Fig. 19 is a flowchart illustrating the generation
5 flow of a temporal user interface.

09763617-014601

DETAILED DESCRIPTION

Figure 1 shows one embodiment of a system

incorporating the present invention. In this embodiment, the

5 system includes a centralized controller 1101 configured to receive information from one or more users from user input device(s) 1114. Also, the centralized controller may receive information from a communications network 1115 through its input/output (I/O) facility 1105, preferably, via a network
10 interface 1107. The I/O facility is capable of both receiving and sending information. Peripheral devices 1113 may be attached to the dynamic analyzer for any number of purposes including, but not limited to: printers for output, scanners for input, additional or alternative storage devices for data
15 storage and retrieval, network interfaces for communication, and devices of the like.

The centralized controller includes a central processing unit (CPU) 1104, random access memory (RAM) 1103, read only memory 1102, and a local storage device 1108. The
20 CPU is electronically coupled to each of the central controller's other elements. The CPU comprises at least one high-speed data processor adequate to execute program modules

for executing user or system generated requests. These

modules are described in Figures 2 through 17. Preferably, the CPU is a conventional microprocessor such as the Intel Pentium Processor. The CPU interacts with RAM, ROM, and

5 storage device(s) to execute stored program code according to conventional data processing techniques.

The local storage device may contain modules. These modules may include, but are not limited to, a dynamic analyzer 1109, a user interface 1110, an operating system
10 1111, a web browser 1112 and a proximity linkage database 1113. These modules may be stored and accessed from the local storage device(s) or from storage devices accessible through I/O. Although these modules typically and preferably are stored in a local storage device, they may also be stored in
15 ROM, RAM, peripheral devices or in remote storage facilities through a communications network.

The operating system is executable program code enabling the operation of a centralized controller. The operating system facilitates access of storage devices, I/O,
20 network interfaces devices, peripheral devices, etc. The operating system preferably is a conventional product such as

09761617-011601

the Unix operating system or Microsoft Windows NT. The

operating system, once executed by the CPU, interacts with
ROM, RAM, I/O, peripheral devices, user input devices, storage
devices, communications networks, program modules, and data,
5 et al. Preferably, the operating system includes
communication protocols that allow the centralized controller
to communicate with other entities through a communications
network. The preferable protocol is TCP/IP.

Figure 2 shows another embodiment of a system
10 incorporating the present invention. In this embodiment, the
centralized controller 1101 embodiment of Figure 1 has been
decentralized into four components: a user interface
controller 2201 or alternatively a user interface device 2202,
a dynamic analyzer controller 2203, a web browser controller
15 2204, a proximity linkage database controller 2205, and a
communications-network navigation-location search engine
(e.g., web search engine, a local crawler search engine, etc.)
2206.

A user interface controller is comprised similarly
20 to the centralized controller of Figure 1 except it does not
require a proximity linkage database, dynamic analyzer, or web

browser. A user interface 2110 is stored program code that is executed by the CPU. The user interface is responsible for receiving either user or system generated requests.

In alternative embodiments, a user interface device
5 2202 may take the place of or be used in conjunction with a user interface controller. The user interface device may be a telephone, a consumer electronics device online access device (e.g., Phillips Inc.'s WebTV), PDA or the like.

A dynamic analyzer controller is comprised similarly
10 to the centralized controller of Figure 1 except it does not require a proximity linkage database, web browser, or user interface. The dynamic analyzer 2109 is stored program code that is executed by the CPU. A dynamic analyzer takes requests from a user interface and provides results to a user
15 interface. The dynamic analyzer may also take system requests.

A proximity linkage database controller is comprised similarly to the centralized controller of Figure 1 except it does not require a dynamic analyzer, web browser, or user
20 interface. A proximity linkage database(s) 2113 is stored program code that is executed by the CPU and it is stored data

processed by the CPU. A proximity linkage database takes

requests from a dynamic analyzer and provides results to a
dynamic analyzer. The proximity linkage database may also
take system requests. In an alternative embodiment, a dynamic
5 analyzer may be integrated into a linkage database or vice
versa, thus, combining the functionality of both. In yet
another alternative embodiment, a dynamic analyzer may be
integrated into a user interface or vice versa, thus,
combining the functionality of both.

10 A web browser controller is comprised similarly to
the centralized controller of Figure 1 except it does not
require a dynamic analyzer, proximity linkage database, or
user interface. A web browser 2112 is stored program code
that is executed by the CPU. Preferably, the web browser is a
15 conventional hypertext viewing application such as Microsoft
Internet Explorer or Netscape Navigator. Preferably, the web
browser allows for the execution of program modules through
facilities such as Java, JavaScript, ActiveX or the like. A
web browser takes requests from a user interface and provides
20 results to a user interface. The web browser may also take
system requests. In alternative embodiments, a web browser

may be integrated into a user interface or vice versa, thus,
combining the functionality of both.

There are several types of web search engines
available. One type is the automated web scanner that has
5 been reading all the information in the World Wide Web (WWW)
and indexing hypertext content for reference in databases;
i.e., monolithic web search engines. Examples include the
likes of Alta Vista, Google, and Yahoo. There are also local
web search engines that run on a user's computer and search
10 the WWW (sometimes referred to as "local crawlers"). Such a
local web search engine may be integrated into or with a
dynamic analyzer, or the dynamic analyzer may communicate with
automated web scanners. Although it is not necessary, it is
preferable to both integrate a web search engine into the
15 dynamic analyzer and to access an external web search engine.
Furthermore, the dynamic analyzer may refer to multiple web
search engines at once, either integrated or remote. Although
some of the above examples reference technologies and web
search engines that index and allow for the search of the
20 Internet, they may be expanded or limited to search other
types of communications networks as well.

The functionality of the user interface controller, dynamic analyzer controller, proximity linkage database controller, web browser controller, and web search engine may be combined in any number of ways to facilitate deployment.

5 To accomplish this, one may simply copy the executable code, first ensuring it has been compiled for the appropriate CPU of the controller for which it is destined, and/or data on to local storage device of one of the various controllers.

09761617 041604
10 Similarly, the functionality of the user interface, dynamic analyzer, proximity linkage database, web browser, and web search engine may be combined in any number of ways to facilitate deployment. To accomplish this, one must simply integrate the components into one code base or in a facility that can dynamically load the components on demand in an
15 integrated fashion.

Figure 3 shows an overview of the basic interaction of the system. The dynamic analyzer 3109 acts as an in-between for: a user interface 3110 on a system, a user interface device 3202, or a web browser 3112 taking requests;
20 and following user actions and enhancing navigation by referring to proximity linkage databases 3113 either directly

or through web search engines 3206. It shows that the dynamic analyzer may service multiple sources at once, and that the dynamic analyzer may access more than one database.

Figure 4 shows web pages 4401 with hypertext 4402 and reference links 4403 at various navigation locations 4404. An originating navigation location 4404a references hypertext that may have initial reference links 4403a. These initial reference links are proximal links to the originating navigation location.

One may view hypertext at an initial reference navigation location 4404b by traversing an initial reference link. The subsequent reference links 4403b found in the hypertext found at the initial reference navigation location are also proximal links, however, they are one reference less proximal (i.e., one "hop" away) to the originating navigation location.

One may view hypertext at a subsequent reference navigation location 4404c by traversing a subsequent reference link. The further subsequent reference links 4403c found in the hypertext found at the subsequent reference navigation location are also proximal links, however, they are two

references less proximal (i.e., two "hops" away) to the

originating navigation location.

Figure 5 outlines the production of dynamic analysis maps. Initially, a user, or even an automated system such as
5 a web bot, navigates a communications network 5501, for example the Internet. Typically this is referred to as "surfing the Internet," "surfing the net," or simply "surfing."

Web browsers and/or the like keep track of the
10 navigation location they are visiting, and can provide this information to other facilities through various application program interfaces (API)s. The dynamic analyzer can thus obtain the current navigation location 5502 the user is visiting through a provided API; for example, Windows Internet
15 Explorer allows this by examining the cache it maintains in a directory or by API. The dynamic analyzer may obtain the navigation location through: a provided API of a web browser, by having it entered directly into the dynamic analyzer, or through another program facility with an API that allows it
20 access to or provides this navigation location.

Upon obtaining the navigation location, the dynamic

analyzer processes the navigation location from the providing source into a format required by a web search engine 5503.

This processing is often simple string and character manipulation formatting navigation location strings into a syntax required for any number of web search engines.

Once the navigation location has been provided, the dynamic analyzer can obtain (i.e., request) related proximal links for the specified navigation location 5504. This request can be made to any number of web search engines. The search engine may be integrated into the dynamic analyzer, or an external web search engine. Upon obtaining the request, the web search engine will perform a search and provide results. The results may be in the form of singular navigation locations, or a subweb. A subweb is a local neighborhood of connected and/or related sites and pages on the Web about a given navigation location.

The dynamic analyzer obtains the request results 5505 from the web search engine. If the web search engine is integrated into the dynamic analyzer, this may be accomplished through: variable passing, object instance variable communication, internal messaging, shared memory space, or the

like. The preferable embodiment will depend on the context of system deployment; i.e., factors such as the capacity of the underlying hardware resources. If the web search engine is external to the data analyzer, capturing the obtained request results may be accomplished through: shared files, process pipes, API information passage, or the like. Again, the preferable embodiment will depend upon the context of system deployment.

Upon having obtained the request results, the dynamic analyzer will process the obtained results 5506. Many times the obtained results, particularly from monolithic web search engines, provided by the web search engine facility include unwanted or poor results: repeated navigation locations, inaccessible navigation locations (i.e., dead links), irrelevant navigation locations, and/or the like. The dynamic analyzer may prune irrelevant and inaccessible navigation locations, rank the results, and otherwise reconcile the results. Such pruning, ranking and reconciling may be accomplished using standard data-processing string, compare, sort techniques, and/or the like. The dynamic analyzer will also examine and rank media type content at the

navigation location such as, but not limited to: size of

textual information; number and size of pictures; the

staleness of the links (i.e., the last time the links were

updated showing that data may be dated and less relevant);

5 number and size of media formats such as, but not limited to,
MP3, AVI, and/or the like; and other types of the like. Such
media format ranking may be determined by user preference or
program preset. Furthermore, the data analyzer will process
the results into a format appropriate for a particular user
10 interface for visualization of the results.

Upon having processed the obtained request results,
the dynamic analyzer provides the processed results to a user
interface for visualization 5507. This processing is often
simple string and character manipulation formatting navigation
15 location strings into any required syntax.

After providing results to the user interface for
visualization, the dynamic analyzer will examine to see if a
termination event has occurred 5508. If a termination event
has not occurred, the dynamic analyzer will again examine how
20 the user navigates a communication network 5501, 5502. If a
termination event has occurred, dynamic analysis mapping

termination results 5509.

Figure 6 outlines a directed search. Initially, a user, or even an automated system such as a web bot, provides a search query 6601. The search query includes, but is not limited to, parameters such as a search subject and a context. Currently, the preferable context is to provide a navigation location. The user may provide the search query to a facility in a user interface, in a hypertext form, or any other facility that can provide the query information to the dynamic analyzer.

The dynamic analyzer obtains the search query 6602. This query may be, but is not limited to being, passed by a user interface, user interface device, a web browser, and others of the like.

Upon obtaining the context, preferably in the form of a navigation location, the dynamic analyzer processes the navigation location from the providing source into a format required by a web search engine 6603. This processing is often simple string and character manipulation formatting navigation location strings into a syntax required by any number of web search engines.

Once the navigation location has been provided, the dynamic analyzer can obtain (i.e., request) related proximal links for the specified navigation location 6604. This request can be made to any number of web search engines. The search engine may be integrated into the dynamic analyzer, or an external web search engine. Upon obtaining the request, the web search engine will perform a search and provide results. The results may be in the form of singular navigation locations, or a subweb.

The dynamic analyzer obtains the request results 6605 from the web search engine. If the web search engine is integrated into the dynamic analyzer, this may be accomplished through: variable passing, object instance variable communication, internal messaging, shared memory space, or the like. The preferable embodiment will depend on the context of system deployment; i.e., factors such as the capacity of the underlying hardware resources. If the web search engine is external to the data analyzer, capturing the obtained request results may be accomplished through: shared files, process pipes, API information passage, or the like. Again, the preferable embodiment will depend upon the context of system

09751617-011601

Upon having obtained the request results, the dynamic analyzer will process the obtained request results 6606. Many times the search results, particularly from 5 monolithic web search engines, provided by the web search engine facility include unwanted or poor results such as: repeated navigation locations, inaccessible navigation locations (i.e., dead links), irrelevant navigation locations, irrelevant subject matter, and/or the like. The dynamic 10 analyzer may prune irrelevant and inaccessible navigation locations, rank the results, and otherwise reconcile the results. Such pruning, ranking and reconciling may be accomplished using standard data-processing string, compare, sort techniques, and/or the like. The dynamic analyzer will 15 also examine and rank media type content at the navigation location such as, but not limited to: size of textual information; number and size of pictures; number of links referencing the information; the staleness of the links (i.e., the last time the links were updated showing that data may be 20 dated and less relevant); number and size of media formats such as, but not limited to, MP3, AVI, and/or the like; and

other types of the like. Such media format ranking may be determined by user preference or program preset. Furthermore, the data analyzer will process the results into a format appropriate for a particular user interface for visualization
5 of the results.

Upon having obtained processed request results, or integrated into the processing of the obtained request results, the dynamic analyzer obtains subject related information in the processed results 6607. Based on the
10 subject matter provided by the user in the search query, the dynamic analyzer ranks the subject matter of the processed results. Such ranking may be accomplished using standard data-processing string, compare, sort techniques, and/or the like. Preferably techniques like a modified shark search or
15 fish search are employed ranking the results' relevancy. The modified search techniques add the novel ability to also examine and rank media type content at the navigation location such as, but not limited to: size of textual information; number and size of pictures; number of links referencing the
20 information; the staleness of the links (i.e., the last time the links were updated showing that data may be dated and less

relevant); number and size of media formats such as, but not limited to, MP3, AVI, and/or the like; and other types of the like. Such media format ranking may be determined by user preference or program preset.

5 After ranking the previous processed request results, the dynamic analyzer may further refine its results by recurrence. The dynamic analyzer can check to see if a search expanse breach has occurred 6608, if not, then the dynamic analyzer may take the ranked processed request
10 results, which preferably have associated navigation locations, and provides the ranked processed request results to the search engine navigation location processor 6603. A search expanse breach will occur based on several factors that may be: preset, or provided by a user or system. A search
15 expanse breach will occur when the search has taken longer than a set amount of time, or has recurred too many times (either in breadth or depth along a graph of subwebs).

 If a search expanse breach does occur, the dynamic analyzer will process the latest search results 6609. Each
20 recurrence, or iteration, of the aforementioned directed search will provide additional search results. These search

results may be maintained in any number of standard data

processing data structures such as lists, arrays, stacks, databases, and/or the like. Preferably a list will maintain the results. Repeated iterations may produce duplicate

5 results, and a range of results of varying relevancy. The final search results are processed removing duplicates and ranking the results based on subject relevancy. Furthermore, the data analyzer will process the results into a format appropriate for a particular output target, e.g., a file, a
10 user interface for visualization of the results, and/or the like.

Upon having processed the obtained final search results, the dynamic analyzer provides the results to a user interface for visualization 6610. This processing is often
15 simple string and character manipulation formatting navigation location strings into any required syntax.

After providing results to the user interface for visualization, the dynamic analyzer will examine to see if a termination event has occurred 6611. If a termination event
20 has not occurred, the dynamic analyzer will again examine users search query 6601, 6602. If a termination event has

occurred, directed search termination results 6612.

Figure 7 outlines a dynamic directed search. It is the same as the directed search of figure 6 except: processed search results 6609 and figure 7 analogue 7708, and provision of results to the user interface 6610, 7709 occur before the search expanse breach check 6608, 7710; the search expanse breach check 7710 now occurs after the provision of results to the user interface 7709. Also, now, after the search expanse breach, the termination event check 7711 occurs. This modification allows for dynamic updating and display of search results.

Figure 8 outlines an interpretive help facility. Initially, a user performs some action 8801 where an interpretive help facility is provided. The current environment state is saved. The environment maybe saved in any number of common ways with standard data processing techniques such as, but not limited to: a file, composite data structures, arrays, stacks, and/or the like. The preferred form is to save (i.e., freeze) an object state as provided in many object-oriented languages such as Java, Smalltalk, C++, Objective C, and/or the like; the preferred implementation

language will depend upon the deployment environment. In

addition to the environment state, the last action taken by
the user may be saved 8802.

After freezing the action object state, preferably
5 the action object state is pushed onto a stack of such action
object states 8803. However, the action object states need
not be saved onto a stack, and may be collected using other
standard data structure types.

After pushing the previous action object state onto
10 the state stack, the interpretive help facility will check if
the user has requested help 8804. If the user has not made a
request for help, the interpretive help facility will continue
to archive action object help states 8801, 8802.

If the user has requested help, the interpretive
15 help system will examine the next frozen action object states
8805. The interpretive help facility will thaw and/or examine
the object states and determine the actions taken by the user
and map the action to an index of help topics using standard
data processing comparison techniques such as, but not limited
20 to string compares. In one non limiting example, when thawing
a frozen object state, that state's method last called method

string is compared to a help index for a match. This thawing

process can be recurred backwards in time from the latest

object states backward in time to the earliest. In one

embodiment, the recursion is limited to the last few states.

- 5 The frozen objects may be stored in various data structures.
Preferred data structure embodiments will depend on the
availability of underlying resources, such as processing
power, memory, and/or the like. In one embodiment, the frozen
objects may be saved in a linked list stack, where traversal
10 through the stack is possible without actually popping items
off the stack. As actions are taken, they would be added to
the stack.

- After the proper help action subject has been
selected, the interpretive help facility will request and
15 display the appropriate help information 8806. The user may
then determine if the provided help information 8806 is
relevant 8806b. In one embodiment, the user would be prompted
with a "Yes" or "No" button prompts asking the user if the
displayed help information is relevant. If the information is
20 not relevant, then the interpretive help facility may recur
further through the stack of frozen object states and examine

the next frozen object state 8805. In an alternative

embodiment, a list may be created for the user to make
selections to go to help information topics in a random access
fashion rather than recurring through the stack. In such a
5 list embodiment, a list display widget would be populated with
the actions saved in the frozen object state stack
representing the user's last actions. Selecting any of the
action items would take the user to help matching the action.

10 If the displayed help information is relevant 8806b,
the help information displayed to the user will allow the user
to undo his last action 8807. After the user reads the
appropriate help pages, the interpretive help facility may
provide the user with a prompt asking if based on the help
information that the user believes he or she made a mistake,
15 and if so, will allow the user to undo their mistake by
engaging a provided "mistake" button and/or "undo" command.
If the user does not wish to undo, then the interpretive help
facility will recur and continue to track user actions 8801,
8802.

20 However, if the user wishes to undo her previous
action, the interactive help system can pop the last action

object state from the state stack and instantiate that state using common object oriented data processing techniques for undo systems 8808. After the reinstantiation of the previous object state, the interpretive help facility will recur and
5 continue to track user actions 8801, 8802. It's important to note that an undo functionality does not have to be a part of the interpretive help facility and so after the provision of help information 8806, the interpretive help facility may recur and continue to track user actions 8801, 8802, but it is
10 preferable to provide the undo facility.

Figure 9 illustrates a system display 9901 presented to a user. The system display shows a traditional graphics user interface employing a desktop metaphor containing user selectable icons 9902 and/or the like; i.e., a desktop
15 operating environment 9903. Preferably, the desktop layer is transformed into a temporal pool 9904. This transformation may be operating system dependant. It may be employed by: accessing provided desktop operating environment APIs such as those offered by the Backgrounds preference facilities of
20 Microsoft Windows 98; by patching the operating system; or like techniques. Alternatively, or in conjunction with the

desktop pool, pools maybe implemented in separate windows, in subviews of a window, or the like. The pool also has a pool bottom 9905. The pool bottom may display advertising banners, movies, photos, and other media format types. The pool bottom contents may also be updated in time based on a user or system specified period. An effect may be applied over the pool bottom contents giving it the appearance of liquid visual distortion using standard fishbowl and diffusion transformation techniques, the likes of which may be seen in applications such as Adobe Photoshop. Although the discussion of the temporal quantum interface will be framed in terms of a liquid pool throughout the disclosure, it is to be understood variations on the theme may be implemented, such as by employing a snowflake like metaphor, and/or the like. The raindrops may be represented as squares and/or other arbitrary shapes. The basics of the interface require a display area (the pool), an the ability to view complex (directed graph) information organized in time in an uncluttered manner one time quantum at a time, and allowing the user to access any particular time quantum at random.

Into any pool of water, information rains. The rain

falls from clouds. Clouds form when information is provided from sources such as, but not limited to, a data analyzer. As clouds form, they fill with crystals. Crystals represent navigation locations that have yet to condense and fall into the pool. The data crystals preferably contain information such as: navigation location, subject, media type information, and/or the like; although less information may be provided, the more information available for each crystal, the more the resulting visualization will be informative. Preferably, the clouds and crystals are implemented as data structures and are invisible to the user. The clouds may be implemented employing any number of standard data processing data structures such as lists, arrays, and/or the like, but the preferable data structure form employs a hashtable. A temporal quantum is the difference in time between temporal quantum 2 161602 of Figure 16 and temporal quantum 1 161601 of Figure 16. A temporal quantum is used for graphical state updates and is set either by user preference or system preset. The temporal quantum determines the rate at which crystals condense into raindrops and fall into a pool.

Once a temporal quantum has transpired, a crystal

will condense and fall into the pool disturbing the water,

thus creating raindrops 101001 of Figure 10. The ordering of

which crystals from an information cloud will form into

raindrops may be determined by the ordering of data in the

5 information cloud data structure. In alternative embodiments,

the order of raindrop formation may be varied based on user

preferences, e.g., media formats are set to form faster than

text data (i.e., media being heavier and/or larger data). The

raindrops disturb the water similarly to how such natural

10 phenomena occurs, thus, causing ripples. The visual rendering

of such an effect is widely known in data processing and may

be achieved by using techniques such as fish bowl and

diffusion transformations, the likes of which may be seen in

popular applications such as Adobe Photoshop.

15 The raindrops fall in groups based on the

relationship of the underlying data they represent. Thus, for

example, if a data analyzer provides a subweb parent page

(i.e., the web page referring to other related web pages) and

this information is crystallized in a cloud and falls into a

20 pool, the main ring will represent the subweb parent page.

The size of a raindrop will be determined by any user or

system specified criteria such as, but not limited to: size of

the document, number of related links, number of photos,

staleness of links, media content type, subject relevancy

ranking, or the like. These criteria may be accessed and

5 modified by the user through a dynamic mapping and search

criteria menu 9906 or like facility, which will show the user

current dynamic visual criteria 9907, and upon selection allow

users to modify the visualization by setting parameters in a

dialogue box or like facility. The preferable default is for

10 the raindrop size to be related to the number of additional

references contained within.

The color of a raindrop will be determined by any of
the aforementioned user or system specified criteria. The

preferable default is for color to vary with the size of the

15 content to be accessed: e.g., red raindrops represent large

documents that might take longer to access, yellow raindrops

represent smaller documents, and green raindrops may represent

small and fast loading documents. Any number of color to

criteria mappings may be employed, however, the preferred

20 default is to limit the granularity of the mappings to the

three aforementioned colors. Similarly, levels of

transparency, and thickness of the raindrop rings may be used to represent any of the aforementioned criteria; however, this is not preferable by default as the visualization may become more complicated and cluttered.

5 Raindrops that fall or appear in proximity to one another represent related information to the user. Raindrops representing related or referenced information, for example the reference links of subweb parent, that fall completely within another raindrop represent a related raindrop with no
10 external references 101002 of Figure 10 pointing to sources outside the domain of the subweb parent. Raindrops that intersect partially inside its parent raindrop, and partially outside, represent a related raindrop with external references 101003 of Figure 10 pointing to sources outside the domain of
15 the related raindrop's parent raindrop and with non external references pointing to sources inside the domain. Raindrops that touch only the outside of a parent raindrop and are otherwise outside the parent raindrop represent a related raindrop with substantially all external references pointing
20 to sources outside the domain. Raindrops that do not touch or intersect are not related. Raindrops that have the same

09761617.011601

raindrop parent and that reference one another will intersect

to show the inter-referencing 101004 of Figure 10. Only the

most relevant raindrop references will be displayed about a

parent raindrop by default. It is possible, though not

5 preferable, to have additional detail displayed, increasing

the number of raindrops about a parent raindrop. Raindrops

within a parent may further represent their own references in

like fashion becoming sub-parents themselves. Subsequent

raindrops related to their sub-parents will appear and may

10 appear to grow in towards the center of the original parent

raindrop, or outwards from the original parent based on user

preference or system preset. However, it is not preferable to

show such recursion as the display may become cluttered.

When raindrops form they are preferably labeled with

15 a title from the data that the raindrop represents, usually a

navigation location title 101005 of Figure 10. When a user

moves a pointing device over a title of a navigation location

it will grow in size and show additional detail. Also, if the

liquid temporal user interface is provided with information

20 that the raindrop should reflect other media types it can

provide alternative and/or complementary dynamic visual cues,

preferably as small icon representations, 9908 such as, but

not limited to: downloadable media content 101006 of Figure

10, which represents that content is available for

downloading; password protected media 111101 of Figure 11,

5 which represents content requiring a password for access;

alerted subject matter media 111102 of Figure 11, which

represents a content that may not be appropriate (e.g., the

likes of which may be flagged by facilities such as NetNanny,

et al.); interactive media content 111103 of Figure 11, which

10 represents content that is dynamic in nature like live stream

feeds, chat rooms, discussion boards, and/or the like; and

notes available media content 111104 of Figure 11, which

represents content that has annotated notes available.

The raindrops dissolve 9910 with time and the water

15 calms 151501 as in Figure 15 until new raindrops appear 161604

as in Figure 16 at the next temporal quantum. The raindrops

will dissolve if the user does not interact with it or the web

page underlying it. The rate of disappearance is predicated

on how frequently the user interacts with them. The time line

20 facility 9909 allows a user to go back and forth through the

time of the data rain storm by selecting and manipulating the

time line knob 9911. Manipulating the time line knob will cause the pool to churn and change its representation to the appropriate state and will cause raindrops to fade 161603 of Figure 16 and form 161604 of Figure 16.

5 At any given time, the user may select a raindrop, usually by clicking the ring with a pointing device. Once a raindrop is selected, the liquid temporal user interface will provide a web browser 9912 with a new navigation location to view. A box will highlight the active context 101007 of
10 Figure 10 representing that the highlighted raindrop(s) are currently being viewed in the web browser. The appropriate set of raindrops will be highlighted as the active context either when the user selects them within the liquid temporal user interface, or whilst the liquid temporal user interface
15 is updated by a dynamic analysis mapping, directed searches, or the like; i.e., following the actions of the web browser or a search facility 9913.

 A search selection facility may be provided, although not required, into which search subjects and other
20 search criteria may be entered. These search query parameters along with the navigation location of the active context are

provided to facilities such as, but not limited to: directed

searches and dynamic directed searches. A search selection facility may be accessed via a menu, a button, a dialogue box, and/or the like.

5 Also, an interpretive help tool 9914 may be provided, although not required. Selecting the interpretive help tool will engage an integrated interpretive help facility providing it with an active context state.

10 In Figure 11 a user is extending a selection marquee about a set of raindrops that are of interest. In Figure 12, after making the selection with the marquee, a detail focus box appears about the selection 121201. The detail focus box may be closed by selecting the close box 121202, windowfied by selecting the windowfy box 121203, semantically magnified (in
15 or out) by selecting the magnification box 121204, or the contents within the detail focus box may be eliminated by selecting the trash box 121205. The creation of a detail focus box automatically increases the level of semantic detail shown within its borders as can be seen by the transformation
20 of the contents within the selection marquee in Figure 11 and the resulting more detailed view within the detail focus box

in Figure 12. The increased level of semantic detail can be

provided by a dynamic analyzer recurring and semantically

increasing the level of detail shown. Figure 13 illustrates

that a user may move a detail focus box about the screen and

5 that any raindrops it covers will increase in the amount of
semantic detail shown 131301.

Figure 14 shows a windowfied view of a pool 141401.

A user may scroll inside the windowfied view, or simply use it

for ease of access. Windowfied views may be saved in a subweb

10 bookmark facility by engaging a save key. Also, elements may
be dragged and dropped into and out from a windowfied view.

Figure 15 shows that selecting the trash box in a detail focus

box 151502 results in the pool water being calmed 151501. The

pool underlying the detail focus box is calmed, i.e., the

15 raindrops in the detail focus box are deleted.

Figure 17 shows a skimming a pebble 171701. A user

can skim a pebble, preferably, by clicking and dragging over

an area in the pool. The display will show a rippling,

modeled after the natural phenomena, until the user stops

20 dragging; i.e., until the user stops the selection or

skimming. Raindrops that the user skims pebbles over will not

dissolve and will remain in view. The liquid temporal user

interface can then inform a data analyzer or web page to cache
the web pages represented by the skimmed raindrops so when a
user subsequently selects a skimmed raindrop, the web browser
5 will display the referenced information more quickly.

Figure 18 outlines a liquid temporal user interface
(LTUI) flow. Initially, the temporal user interface is loaded
an initialized so that it may be engaged 181801. Thereafter,
a user may engage the user interface 181802. The user may
10 engage the LTUI by interacting with any number of conventional
user peripheral devices such as directing a pointer through a
device, e.g., a mouse. As the user interacts with the LTUI,
the LTUI checks for user selections 181803, e.g., checking for
user drags for creating ripples. Thereafter, the LTUI will
15 update its internal state depending upon any selections
181804. Upon updating its state, the LTUI will redraw the
display reflecting the updated state 181805. Thereafter, the
LTUI may check for a termination event, e.g., the user
requesting to quit the LTUI 181805. If there is no
20 termination event then iteration will continue with the LTUI
checking for any new user selections 181803. If there is a

termination event 181805, then the LTUI will save its state

181806. The LTUI may employ standard data structures, such as

custom structs, arrays, lists, and frozen objects. Upon

saving the LTUI state 181806, the LTUI clears its state 181807

5 and redraws the screen 181808 so as to remove its display

elements from the screen and thereafter terminate 181809.

Figure 18 outlines the display flow of liquid

temporal user interface (LTUI). Initially, the temporal user

interface is loaded and initialized so that it may be engaged

10 181801. Thereafter, a user may engage the user interface

181802. The user may engage the LTUI by interacting with any

number of conventional user peripheral devices such as

directing a pointer through a device, e.g., a mouse. As the

user interacts with the LTUI, the LTUI checks for user

15 selections 181803, e.g., checking for user drags for creating

ripples. Thereafter, the LTUI will update its internal state

depending upon any selections 181804. Upon updating its

state, the LTUI will redraw the display reflecting the updated

state 181805. Thereafter, the LTUI may check for a

20 termination event, e.g., the user requesting to quit the LTUI

181805. If there is no termination event then iteration will

continue with the LTUI checking for any new user selections

181803. If there is a termination event 181805, then the LTUI will save its state 181806. The LTUI may employ standard data structures, such as custom structs, arrays, lists, and frozen
5 objects. Upon saving the LTUI state 181806, the LTUI clears its state 181807 and redraws the screen 181808 so as to remove its display elements from the screen and thereafter terminate 181809.

Figure 19 outlines the generation flow of liquid
10 temporal user interface. Initially, the temporal user interface is loaded and initialized so that it may be engaged 191901. Thereafter, the LTUI is displayed 191905 and Figure 18. Next, the LTUI obtains information for temporal display from a data source 19602. Although the temporal information
15 may be obtained from any number of data sources, in one non-limiting embodiment, the data source is a data analyzer. Next the LTUI may generate information clouds including the information obtained from the data source 191903. In an alternative embodiment, the LTUI may simply reference the data
20 from the data source. The LTUI passes the information from the data source into a data structure, e.g., a hashtable.

Next, the LTUI generates information crystals referencing

information in the information clouds that will 191904 that
will be used to form raindrops. Formed crystals are then
rendered as raindrops at the passing of a temporal quantum by
5 having the LTUI redraw 191905 and Figure 18. Thereafter, the
LTUI may check for a termination event, e.g., the user
requesting to quit the LTUI 191906. If there is no
termination event then iteration will continue with the LTUI
obtaining updated information from a data source 19602. If
10 there is a termination event 191906, then the LTUI will
redraw, and as a consequence save its state, 191905 and Figure
18 and thereafter terminate 191909.

The user may engage the LTUI by interacting with any
number of user conventional peripheral devices such as
15 directing a pointer through a device, e.g., a mouse. As the
user interacts with the LTUI, the LTUI checks for user
selections 181803, e.g., checking for user drags for creating
ripples. Thereafter, the LTUI will update its internal state
depending upon any selections 181804. Upon updating its
20 state, the LTUI will redraw the display reflecting the updated
state 181805. Thereafter, the LTUI may check for a

termination event, e.g., the user requesting to quit the LTUI

181805. If there is no termination event then iteration will continue with the LTUI checking for any new user selections

181803. If there is a termination event 181805, then the LTUI

5 will save its state 181806. The LTUI may employ standard data structures, such as custom structs, arrays, lists, and frozen objects. Upon saving the LTUI state 181806, the LTUI clears its state 181807 and redraws the screen 181808 so as to remove its display elements from the screen and thereafter terminate
10 181809.

It should be understood that the above description
15 is only representative of illustrative embodiments. For the convenience of the reader, the above descriptions have focused on a representative sample of all possible embodiments, a sample that teaches the principles of the invention. The description has not attempted to exhaustively enumerate all
20 possible variations. That alternate embodiments may not have been presented for a specific portion of the invention or that

further undescribed alternate embodiments may be available for

a portion is not to be considered a disclaimer of those

alternate embodiments. It will be appreciated that many of

those undescribed embodiments incorporate the same principles

5 of the invention and others are equivalent. Thus, it is to be

understood that the embodiments and variations shown and

described herein are merely illustrative of the principles of

this invention and that various modifications may be

implemented without departing from the scope and spirit of the

10 invention.

09761617-011601